GUÍA DE REFERENCIA TÉCNICA
VS Code · SQLTools

Snippets · IntelliSense · Funciones y Métodos de Uso

MySQL 8.4.8 (WSL2) | SQL Server 2022 | MOOC_PRACTICA_ORM

Emilio Fernández Abalde
Entorno: Ubuntu 26.04 LTS (WSL2) · VS Code · SQLTools Extension
Abril 2026

1. Introducción a SQLTools para VS Code

SQLTools es una extensión de código abierto para Visual Studio Code que centraliza la gestión de bases de datos relacionales y NoSQL directamente en el editor. Su propósito fundamental es eliminar la necesidad de alternar entre el editor de código y clientes externos de bases de datos, ofreciendo una experiencia integrada de desarrollo de consultas, exploración de esquemas y ejecución de scripts SQL.

En el contexto del proyecto MOOC_PRACTICA_ORM, SQLTools actúa como herramienta complementaria al stack Node.js/Express/Sequelize v6, permitiendo verificar el estado del esquema relacional, depurar consultas generadas por el ORM y ejecutar scripts de migración directamente desde el entorno de desarrollo.

1.1. Componentes Principales

La extensión se estructura en tres módulos funcionales diferenciados:

- SQLTools Core: Motor central que gestiona las conexiones, el historial de consultas y la interfaz gráfica del explorador de base de datos.
- Drivers (Controladores): Paquetes independientes que implementan la conectividad con cada motor de base de datos. Para el entorno actual se requieren el driver para MySQL y el driver para SQL Server (mssql).
- Language Server: Servidor de lenguaje que alimenta el autocompletado IntelliSense SQL en el editor de código.

1.2. Requisitos del Entorno

Componente	Versión / Valor	Estado
VS Code	≥ 1.80 (recomendado 1.90+)	□ Instalado

Componente	Versión / Valor	Estado
SQLTools Extension	0.28.x	✅ Activo
Driver MySQL	sqltools-driver-mysql	✅ Configurado
Driver SQL Server	sqltools-driver-mssql	✅ Configurado
MySQL	8.4.8 (WSL2 localhost:3306)	✅ Disponible
SQL Server	2022 (172.25.102.202:1433)	✅ Disponible
Node.js	v22.20.0	✅ Activo

2. Snippets SQL — Referencia Completa

Los snippets son plantillas de código predefinidas que SQLTools inyecta en el editor mediante autocompletado. Se activan escribiendo el prefijo del snippet en un archivo .sql y seleccionando la opción del menú desplegable de IntelliSense (visible en la Imagen 1 de referencia). Todos los snippets admiten posicionamiento de cursor mediante tabulación entre los marcadores de posición (placeholders).

2.1. Catálogo de Snippets Disponibles

Snippet	Función	Descripción Extendida
sqlAddColumn	Añadir columna	Genera ALTER TABLE ... ADD COLUMN con tipo de dato, restricciones NOT NULL y valor DEFAULT configurables mediante placeholders.
sqlCreateDatabase	Crear base de datos	Emite CREATE DATABASE con soporte para CHARACTER SET y COLLATE en MySQL, y COLLATION en SQL Server.
sqlCreateStoredProc	Crear procedimiento almacenado	Genera la estructura completa CREATE PROCEDURE con bloque BEGIN...END, parámetros IN/OUT y comentarios de documentación.
sqlCreateTable	Crear tabla	Plantilla CREATE TABLE con PRIMARY KEY, tipos de datos comunes, NOT NULL y ENGINE=InnoDB (MySQL) o WITHOUT ROWLOCK (SQL Server).
sqlCreateView	Crear vista	Genera CREATE OR REPLACE VIEW con SELECT configurable, útil para abstracción de consultas complejas en el ORM.
sqlDeleteRows	Eliminar filas	Produce DELETE FROM ... WHERE con marcador de posición para la condición. Incluye advertencia de seguridad en el comentario.
sqlDropColumn	Eliminar columna	Genera ALTER TABLE ... DROP COLUMN de forma segura.
sqlDropDatabase	Eliminar base de datos	Emite DROP DATABASE IF EXISTS, evitando error si la base de datos no existe.
sqlDropTable	Eliminar tabla	Genera DROP TABLE IF EXISTS con opción CASCADE en SQL Server.
sqlDropView	Eliminar vista	Produce DROP VIEW IF EXISTS.
sqlExtensionHelp	Ayuda de	Abre la documentación integrada de SQLTools con

Snippet	Función	Descripción Extendida
	extensión	atajos de teclado y comandos disponibles.
sqlGetSpaceUsed	Espacio en disco	Ejecuta sp_spaceused (SQL Server) o consulta INFORMATION_SCHEMA.TABLES (MySQL) para mostrar el espacio consumido por tablas.

2.2. Sintaxis Generada por Snippet

2.2.1. sqlAddColumn

Uso: añadir una columna a una tabla existente sin recrearla.

```
ALTER TABLE nombre_tabla
ADD COLUMN nombre_columna TIPO_DATO [NOT NULL] [DEFAULT valor];
```

Ejemplo práctico (orm_bbdd):

```
ALTER TABLE empleados
ADD COLUMN fecha_baja DATE NULL DEFAULT NULL;
```

2.2.2. sqlCreateDatabase

```
-- MySQL
CREATE DATABASE IF NOT EXISTS nombre_bbdd
  CHARACTER SET utf8mb4
  COLLATE utf8mb4_unicode_ci;
```

```
-- SQL Server
CREATE DATABASE nombre_bbdd
  COLLATE Latin1_General_CI_AS;
```

2.2.3. sqlCreateStoredProc

```
-- MySQL
DELIMITER //
CREATE PROCEDURE nombre_proc(IN param1 INT, OUT param2 VARCHAR(100))
BEGIN
  -- Lógica del procedimiento
  SELECT columna INTO param2 FROM tabla WHERE id = param1;
END //
DELIMITER ;
```

2.2.4. sqlCreateTable

```
CREATE TABLE IF NOT EXISTS nombre_tabla (
  id         INT          NOT NULL AUTO_INCREMENT,
  nombre     VARCHAR(100) NOT NULL,
  created_at DATETIME     NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

2.2.5. sqlCreateView

```
CREATE OR REPLACE VIEW v_nombre AS
SELECT col1, col2
FROM   tabla_base
WHERE  condicion = valor;
```

2.2.6. sqlDeleteRows

```
-- ⚠ PRECAUCIÓN: sin WHERE elimina TODOS los registros
DELETE FROM nombre_tabla
```

```
WHERE  condicion = valor;
```

ℹ Precaución: ejecutar siempre DELETE precedido de un SELECT equivalente para verificar el conjunto afectado.

2.2.7. sqlDropColumn / sqlDropTable / sqlDropView / sqlDropDatabase

```
ALTER TABLE nombre_tabla DROP COLUMN nombre_columna;
DROP TABLE  IF EXISTS nombre_tabla;
DROP VIEW   IF EXISTS nombre_vista;
DROP DATABASE IF EXISTS nombre_bbdd;
```

2.2.8. sqlGetSpaceUsed

```
-- MySQL
SELECT table_name, ROUND((data_length + index_length)/1024/1024, 2) AS MB
FROM   information_schema.TABLES
WHERE  table_schema = 'orm_bbdd'
ORDER  BY (data_length + index_length) DESC;
```

```
-- SQL Server
EXEC sp_spaceused 'nombre_tabla';
```

3. IntelliSense SQL — Autocompletado Contextual

SQLTools implementa un servidor de lenguaje LSP (Language Server Protocol) que analiza el contexto sintáctico de la consulta en edición y ofrece sugerencias de palabras clave SQL, nombres de tablas, columnas y esquemas en tiempo real. Las opciones visibles en la Imagen 2 corresponden a las cláusulas que pueden suceder a FROM en una instrucción SELECT.

3.1. Cláusulas Sugeridas tras FROM — Referencia

Cláusula	Tipo JOIN	Resultado del Conjunto
AS	Alias	Asigna un nombre alternativo a tabla o columna. No filtra ni combina filas.
JOIN / INNER JOIN	Intersección	Devuelve únicamente las filas que tienen correspondencia en ambas tablas.
LEFT JOIN / LEFT OUTER JOIN	Exterior izquierdo	Devuelve todas las filas de la tabla izquierda y las coincidentes de la derecha; NULL donde no hay coincidencia.
RIGHT JOIN / RIGHT OUTER JOIN	Exterior derecho	Devuelve todas las filas de la tabla derecha y las coincidentes de la izquierda; NULL donde no hay coincidencia.
FULL JOIN / FULL OUTER JOIN	Exterior completo	Combina LEFT y RIGHT JOIN. Devuelve todas las filas de ambas tablas; NULL en las no coincidentes. No disponible en MySQL nativo (se emula con UNION).
WHERE	Filtro	Filtra filas individuales antes de la agrupación. Actúa sobre columnas base, no sobre agregados.

Cláusula	Tipo JOIN	Resultado del Conjunto
GROUP BY	Agrupación	Agrupa filas con valores iguales en las columnas especificadas para aplicar funciones de agregado (COUNT, SUM, AVG...).
HAVING	Filtro de grupo	Filtra grupos resultantes de GROUP BY. Permite usar funciones de agregado en la condición, a diferencia de WHERE.

3.2. Diagrama de Tipos JOIN

La siguiente tabla ilustra el resultado de cada tipo de JOIN con un ejemplo concreto sobre las tablas de orm_bbdd:

Tipo	Filas retornadas	Nulos posibles
INNER JOIN	Solo coincidencias A ∩ B	Ninguno
LEFT JOIN	Todo A + coincidencias B	Columnas de B
RIGHT JOIN	Todo B + coincidencias A	Columnas de A
FULL OUTER JOIN	Todo A ∪ B	Columnas de A y de B

3.3. Ejemplos Prácticos sobre orm_bbdd

3.3.1. INNER JOIN — pedidos con cliente existente

```
SELECT p.id, p.fecha, c.nombre AS cliente
FROM   pedidos AS p
INNER  JOIN clientes AS c ON p.cliente_id = c.id
WHERE  p.estado = 'completado'
ORDER  BY p.fecha DESC;
```

3.3.2. LEFT JOIN — todos los clientes, con o sin pedido

```
SELECT c.id, c.nombre, COUNT(p.id) AS total_pedidos
FROM   clientes AS c
LEFT   JOIN pedidos AS p ON c.id = p.cliente_id
GROUP  BY c.id, c.nombre
HAVING COUNT(p.id) = 0;  -- Clientes sin ningún pedido
```

3.3.3. FULL OUTER JOIN (MySQL — emulación con UNION)

```
SELECT c.id, c.nombre, p.id AS pedido_id
FROM   clientes c LEFT  JOIN pedidos p ON c.id = p.cliente_id
UNION
SELECT c.id, c.nombre, p.id AS pedido_id
FROM   clientes c RIGHT JOIN pedidos p ON c.id = p.cliente_id;
```

ℹ MySQL 8.x no implementa FULL OUTER JOIN de forma nativa. La emulación mediante UNION ALL con condición IS NULL es la práctica estándar.

3.3.4. GROUP BY + HAVING — ventas por categoría

```
SELECT   cat.nombre AS categoria,
         COUNT(p.id) AS num_pedidos,
         SUM(p.total) AS ingresos
FROM     pedidos AS p
INNER    JOIN categorias AS cat ON p.categoria_id = cat.id
GROUP BY cat.nombre
HAVING   SUM(p.total) > 1000
```

```
ORDER BY ingresos DESC;
```

4. Configuración de Conexiones en SQLTools

Las conexiones en SQLTools se definen en el archivo settings.json del workspace o del usuario de VS Code, bajo la clave sqltools.connections. Cada entrada del array corresponde a una conexión independiente.

4.1. Conexión MySQL 8.4.8 (WSL2)

```
"sqltools.connections": [
  {
    "name":     "MySQL - orm_bbdd (WSL2)",
    "driver":   "MySQL",
    "server":   "localhost",
    "port":     3306,
    "database": "orm_bbdd",
    "username": "Emilio_Fernadez_Abalde",
    "askForPassword": true
  },
```

4.2. Conexión SQL Server 2022

```
  {
    "name":     "SQL Server 2022 - orm_bbdd",
    "driver":   "MSSQL",
    "server":   "172.25.102.202",
    "port":     1433,
    "database": "orm_bbdd",
    "username": "sa",
    "askForPassword": true,
    "trustServerCertificate": true
  }
]
```

ℹ trustServerCertificate: true es necesario en entornos de desarrollo con SQL Server sin certificado TLS firmado por CA reconocida.

4.3. Atajos de Teclado Esenciales

Acción	Atajo (Linux/WSL2)	Descripción
Abrir conexión	Ctrl + Alt + E	Despliega el panel de conexiones de SQLTools.
Ejecutar consulta seleccionada	Ctrl + E, Ctrl + E	Ejecuta solo el texto seleccionado o la sentencia actual.
Ejecutar todo el archivo	Ctrl + E, Ctrl + A	Ejecuta el archivo .sql completo en la conexión activa.
Abrir nueva query	Ctrl + Alt + N	Crea un nuevo archivo temporal de consulta.
Historial de consultas	Ctrl + E, Ctrl + H	Muestra el historial de consultas ejecutadas.
Autocompletado	Ctrl + Space	Fuerza la apertura del menú IntelliSense

Acción	Atajo (Linux/WSL2)	Descripción
		en posición actual.
Formatear SQL	Ctrl + Shift + P → Format	Aplica indentado y mayúsculas a palabras clave SQL.

5. Integración con el Stack Sequelize v6 + Node.js v22

En el flujo de desarrollo con ORM, SQLTools cumple un papel de complemento de verificación y depuración. Las migraciones generadas por sequelize-cli producen SQL que puede inspeccionarse y ejecutarse directamente en SQLTools para validar el esquema antes o después de aplicar la migración programática.

5.1. Flujo de Trabajo Recomendado

Fase	Herramienta	Acción
1. Diseño de modelo	Sequelize Model	Definir atributos, tipos y asociaciones en el modelo JS/TS.
2. Generación de migración	sequelize-cli	npx sequelize-cli migration:generate --name crear-tabla-X
3. Revisión SQL	SQLTools + snippet sqlCreateTable	Verificar el DDL equivalente antes de ejecutar la migración.
4. Ejecución	sequelize-cli	npx sequelize-cli db:migrate
5. Validación	SQLTools + sqlGetSpaceUsed	Confirmar que la tabla existe y tiene el esquema correcto.
6. Testing de consultas	SQLTools IntelliSense	Prototipar JOINs y filtros antes de implementarlos en Sequelize.

5.2. Equivalencias ORM ↔ SQL

Sequelize v6 (JS)	SQL Equivalente (SQLTools)
Model.findAll({ where: { id: 1 } })	SELECT * FROM tabla WHERE id = 1;
Model.findAll({ include: [Asociacion] })	SELECT ... FROM tabla INNER JOIN otra ON ...;
Model.count({ group: 'columna' })	SELECT columna, COUNT(*) FROM tabla GROUP BY columna;
Model.create({ col: val })	INSERT INTO tabla (col) VALUES (val);
Model.update({ col: val }, { where })	UPDATE tabla SET col = val WHERE ...;

Sequelize v6 (JS)	SQL Equivalente (SQLTools)
Model.destroy({ where: { id: 1 } })	DELETE FROM tabla WHERE id = 1;

ℹ Sequelize genera SQL con alias automáticos y comillas de escape. Las consultas vistas en SQLTools pueden diferir en formato pero son semánticamente equivalentes.

6. Buenas Prácticas y Consideraciones de Seguridad

6.1. Uso Seguro de Snippets Destructivos

Los snippets sqlDeleteRows, sqlDropTable, sqlDropDatabase y sqlDropColumn generan operaciones DDL/DML destructivas. Las siguientes prácticas son obligatorias en entornos de desarrollo compartido:

- Ejecutar siempre SELECT equivalente antes del DELETE para verificar el subconjunto afectado.
- Envolver operaciones destructivas en una transacción explícita (BEGIN TRANSACTION / ROLLBACK en SQL Server; START TRANSACTION / ROLLBACK en MySQL).
- Confirmar la conexión activa antes de ejecutar: el indicador de base de datos activa aparece en la barra inferior de VS Code.
- Nunca ejecutar DROP DATABASE sobre orm_bbdd o employees sin respaldo previo.

6.2. Variables de Entorno y Credenciales

Las credenciales de conexión no deben estar en texto plano en settings.json del repositorio. La configuración askForPassword: true de SQLTools es el mecanismo correcto para entornos de desarrollo. Para automatización CI/CD utilizar variables de entorno gestionadas por dotenv.

6.3. Diferencias MySQL vs SQL Server en Sintaxis

Operación	MySQL 8.4.8	SQL Server 2022
Limitar filas	LIMIT n	TOP n / FETCH NEXT n ROWS ONLY
Auto-incremento	AUTO_INCREMENT	IDENTITY(1,1)
Concatenar texto	CONCAT() / operador ||	CONCAT() / operador +
Fecha actual	NOW() / CURDATE()	GETDATE() / GETUTCDATE()
Full Outer Join	Emular con UNION	FULL OUTER JOIN nativo
Comillas de identificador	Backtick (`)	Corchetes [] o comillas dobles
Tipo texto variable	VARCHAR(n) UTF8MB4	NVARCHAR(n) Unicode nativo

Glosario de Términos

Término	Definición
Snippet	Plantilla de código reutilizable activada mediante un prefijo en el editor de texto.
IntelliSense	Sistema de autocompletado contextual implementado por VS Code mediante el protocolo LSP.
DDL	Data Definition Language — instrucciones SQL que definen la estructura (CREATE, ALTER, DROP).
DML	Data Manipulation Language — instrucciones que manipulan datos (SELECT, INSERT, UPDATE, DELETE).
JOIN	Operación relacional que combina filas de dos o más tablas según una condición de enlace.
ORM	Object-Relational Mapping — capa de abstracción que mapea objetos de código a tablas relacionales.
LSP	Language Server Protocol — protocolo estándar para comunicar editores con servidores de análisis de código.
Driver SQLTools	Plugin independiente que implementa el protocolo de comunicación de SQLTools para un motor de BD concreto.